advanced
People. Partnership. Solutions.

Automated Refactoring of a New York Times Mainframe to AWS with Advanced

This case study describes the project, including the automated refactoring process and AWS architecture, as well as key lessons learned, business outcomes, and future technology plans.

The New York Times had a critical business workload running on a mainframe as the core IT system supporting the daily Home Delivery Platform of its newspaper. They collaborated with Advanced, an AWS Partner Network (APN) Select Technology Partner, to successfully transform their legacy COBOL-based application into a modern Java-based application, which today runs on Amazon Web Services (AWS).

Using innovative automated refactoring, the application was modernized to object-oriented code, and the data was transformed from legacy indexed-files to a relational database.

The New York Times Context and Objectives

The New York Times is an American newspaper with worldwide influence and readership. Founded in 1851, the paper has won 125 Pulitzer Prizes, more than any other newspaper. The Times is ranked 17th in the world by circulation and second in the United States.

The company's core-business application managed daily home delivery of the newspaper since 1979, supporting a line of business worth more than $500 million annually. It represented years of accumulated experience and knowledge, and yet it significantly resisted modification and evolution.

In addition, the IBM Z mainframe running the z/OS operating system was expensive to operate in comparison to more modern platforms that had evolved at the company. It needed modernization to reduce operating costs and enable the convergence of the Digital Platform with the Home Delivery Platform.

An attempt to manually rewrite the home delivery application between 2006 and 2009 failed. In 2015, an evaluation of alternate approaches determined that a second attempt at redeveloping the application would have been much more expensive, and an alternative emulator rehosting would have continued to lock-up data in proprietary technology.

With mounting pressure to quickly lower costs, the chosen strategy was to migrate code and data with automated refactoring. This approach promised functional equivalence, lower operational cost, and easier integration with modern technologies.

Client >
New York Times

Sector >
Mass Media

Project >
Legacy COBOL-based application to a modern Java-based application, which runs on Amazon Web Services (AWS)

Automated Refactoring of a New York Times Mainframe to AWS with Advanced

Source Mainframe Workload

The mainframe application, named CIS on the mainframe and rebranded to Aristo after the migration, executed business-critical functionality such as billing, invoicing, customer accounts, delivery routing, product catalog, pricing, and financial reporting.

CIS was a z/OS-based CICS/COBOL application with a BMS-based 3270 interface accessing VSAM KSDS business data. Batch processing was supported by JCL jobs with CA7 for job scheduling.

In 2015, CIS had grown to more than 2 million lines of COBOL code, 600 batch jobs, and 3,500 files sent daily to downstream consumers and systems. It consumed around 3 TB of hot data made up of 2 TB of VSAM files, and 1 TB of QSAM sequential files. It used 20 TB of backup cold storage.

Migration with Automated Refactoring

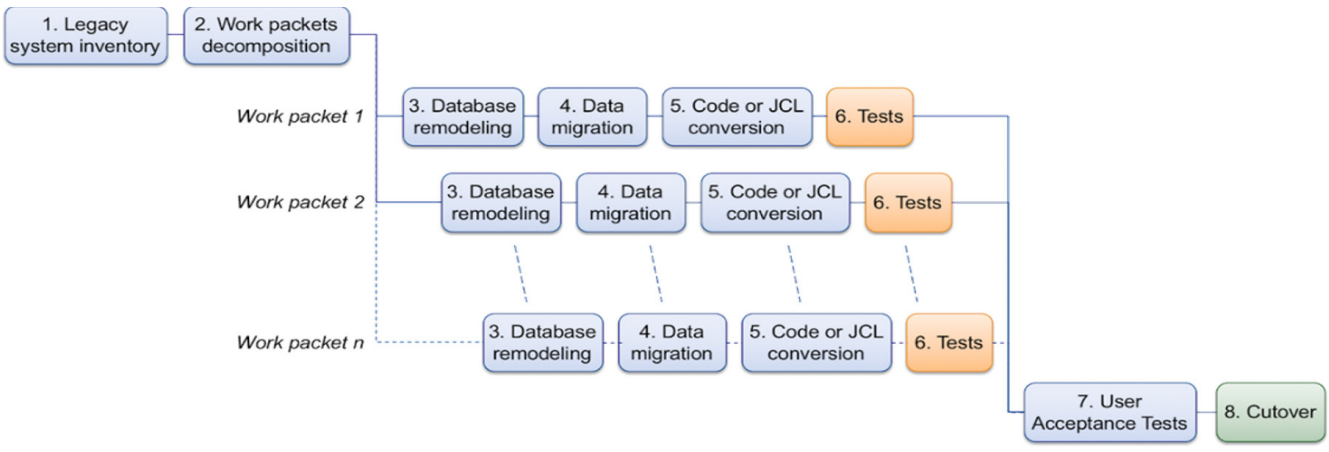
The New York Times selected an automated refactoring approach which retains functional equivalence and critical business logic while converting core legacy applications to maintainable, migrated object-oriented Java. The code is analyzed during the assessment to determine cloud-readiness and the required effort to obtain the desired level of elasticity (i.e. horizontal scalability and vertical scalability) required by the application workloads.

Our COBOL-to-Universal (CTU) software solution supports typical mainframe-based COBOL application components, including CICS, JCL, and common utilities, such as IDCAMS and SORT, in addition to data stores like DB2 for z/OS, IMS, VSAM, and IDMS databases.

The New York Times used CTU and followed an 8-step methodology:

- Step 1 performs an automated inventory of the mainframe and populates a repository of components to be migrated.
- Step 2 consists of a detailed analysis of the applications, data model, architecture preferences, coding styles, database connections, error handling, and refactoring options. All of this leads to the definition of how to piecemeal the code transformation with work packets and the overall test strategy.
- In Step 3, for each work packet, the data model is defined and created in the target database.
- Step 4 automatically generates programs and processes for unloading, transforming, validating, and loading of data from the source data store to the target database.
- In step 5, our CTU software is used to reverse-engineer the COBOL code into an intermediate language, and then to forward-engineer the target Java code.
- Step 6 performs regression tests for each work packet, making sure there is functional equivalence between the source mainframe programs and the new Java code.
- Step 7 is the user acceptance test execution process.
- In Step 8, once these tests are successful, the cutover to production takes place.

Figure 1 - Automated Refactoring Steps



Automated Refactoring of a New York Times Mainframe to AWS with Advanced

Using our CTU software, the resulting Java application became object-oriented and separated into three layers: presentation logic, business logic, and data access.

To maintain the same user experience, CICS BMS Maps were migrated to equivalent web pages which mimic the original 3270 screens as closely as possible. Each COBOL program was refactored to a Java class, and JCL was converted to JSR-352 XML using the Spring Batch runtime for Java. VSAM KSDS files were migrated to a relational Oracle database. During the refactoring process, all VSAM records were analyzed and a DDL generated for each of the best layouts chosen.

The table in Figure 2 shows the technology mapping between the legacy mainframe stack and the target AWS stack.

Replacement components were developed in situations where legacy application dependencies were not supported by the Advanced toolset (e.g. REXX, GVEXPORT), when off-the-shelf software packages were not available as a substitute, or if it made more sense to make use of capabilities of the new environment (vendor database backups and restore points, file system snapshots, etc.).

Functional Equivalence Testing

It was critical to have functional equivalence between the Java application and the COBOL application. Component groups assembled related components that would be runnable and testable together as a single entity through existing externally accessible interfaces, such as web services, user interface screens, database tables, and files.

Technology	Existing Legacy	Target Modernized
Programming language	COBOL	Java
Database	VSAM KSDS files	Oracle
Batch	JCL Jobs/Procs, JCL utilities	Spring Batch XML (JSR-352)
User Interface Screens	CICS BMS Maps	JSF, HTML/CSS, JavaScript
Security and Access Control	RACF	Spring security and Microsoft Active Directory
Middleware, Web Services, Database Transactions	CICS	Jetty, Apache-CXF, JPA/EclipseLink, Modern Systems framework
Reporting	QMF and DB2	Jasper Reports and Oracle
Encryption	Megacryption	Java Cryptography Architecture
Screen Automation	IBM HATS	SOAP Web Services
Batch Process Scheduling	CA-7	BMC Control-M
Monitoring and Alerting	RMF, SMF, Omegamon	New Relic, Nagios, Sumologic
Development and Deployment	Changeman	Git, Gradle, Jenkins, Puppet, Ansible

Figure 2 - Source and target technology mapping

Automated Refactoring of a New York Times Mainframe to AWS with Advanced

Testing accounted for approximately 70-80 percent of the time spent on the project. The testing process was broken down into stages, with each stage progressively increasing in scope and level of difficulty in isolating the root-cause of test failures.of capabilities of the new environment (vendor database backups and restore points, file system snapshots, etc.).

- Stage 1 Pre-Delivery Test: Performed by Modern Systems prior to delivering the refactored code.
- Stage 2 Data Migration Validation: Verified the data is the same between the source VSAM and the target relational database.
- Stage 3 Component Group Test: Verified the functional behavior with one batch job, one or more screen, one Web Service call.
- Stage 4 Batch Process Regression Test: Using static test data (the exact same test data every day) to verify the end-to-end batch process and perform regression testing.
- Stage 5 Batch Process Comparison Test: Using dynamic test data to verify the end-to-end batch with the current day data, and comparing the output between legacy and modernized systems.
- Stage 6 Batch Process Performance Test: Using production data.
- Stage 7 System Integration: This was done in collaboration with other teams and systems within the organization. New transactions are entered via client systems, processed by the batch, and flow to downstream consumers via reports and file feeds.

For these stages, test coverage needed to be high. It can be very time consuming and complex to create test cases, especially for batch jobs. Automation was critical to launch and analyze test cases repeatedly and rapidly.

Target AWS Architecture

As the project progressed, The New York Times pivoted on its overall data center strategy to make Cloud the preferred deployment environment. After less than a year of running in a private data center, Aristo was migrated to AWS. The team had gained significant knowledge of what a successful migration looked like, enabling a migration to AWS with minimal impact to the business.

As shown in Figure 3, once migrated to AWS, the system was broken up into four main components:

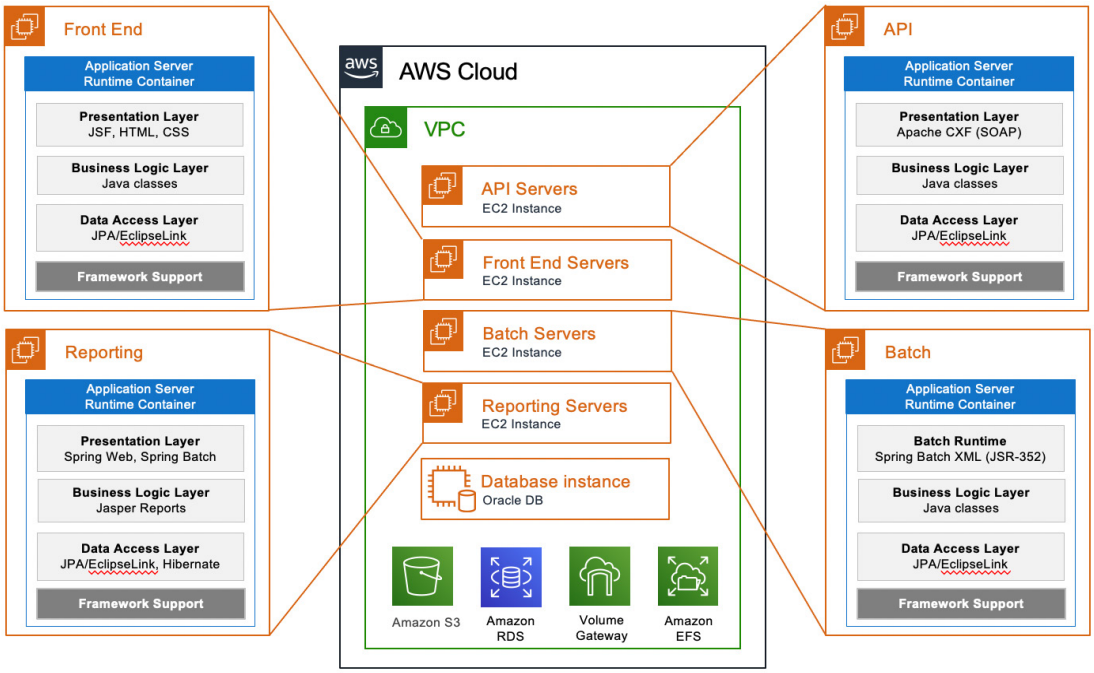


Figure 3 - Target Aristo AWS architecture components

Automated Refactoring of a New York Times Mainframe to AWS with Advanced

- Front End system provides internal operators the ability to manage home delivery subscriptions.
- API system provides SOAP web services to other systems.
- Reporting system builds reports for finance department.
- Batch is the main system where all of the nightly jobs execute. The jobs can run on any instance and the source and destination of the job data is stored on Amazon Elastic File System (Amazon EFS).

The Both the API and Front End systems are within Auto Scaling Groups, providing the ability to respond to a large number of requests. In steady state, the API system uses four m5.xlarge instances and the Front End system uses two m5.xlarge instances. Reporting also uses two m5.xlarge instances. The Batch system is much larger with three m5.4xlarge instances due to the heavy computation needed to run the jobs.

In order to speed up delivery of releases, a Continuous Integration and Continuous Delivery (CI/CD) pipeline was created including:

- Gradle for building and packaging artifacts
- Artifactory for storage and promotion of artifacts
- Jenkins for deployment and orchestration
- Ansible for configuration management
- Hashicorp Vault for secrets management

Migration Timeline

The transformation of the application powering the Home Delivery Platform began in 2015. It was a two-phase process.

Automated Refactoring

This phase lasted around two years and included both the COBOL-to-Java transformation as well as the VSAM-to-relational database conversion, resulting in the Aristo application being launched in production on-premises.

Around the end of this phase, The New York Times announced its cloud strategy impacting the future of Aristo platform and triggering the next phase.

AWS Migration and Enhancements

Once the application was tested and stabilized, the work began in August 2017 to move the application to the AWS Cloud. This was an 8-month project, which also included the following changes:

- From Oracle RAC to Oracle EE
- From Isilon to EFS
- Upgraded Control-M from version 7 to version 8
- Upgraded from FTP to SFTP/S3
- Rebuilt CI/CD pipeline (from Puppet to Ansible)

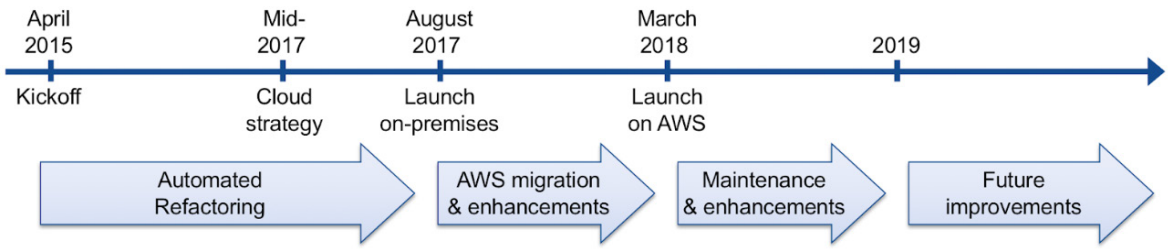


Figure 4 - Mainframe to AWS migration timeline

Automated Refactoring of a New York Times Mainframe to AWS with Advanced

AWS Migration and Enhancements

Once in production on AWS in March 2018, Aristo benefited from maintenance and enhancements including promotion code table expansion, premium Home Delivery (HD) with new digital and paper offerings, and AWS cost optimizations.

Looking ahead, The New York Times is focused on these future improvements:

- Breaking down the application monolith into microservices.
- Continuing convergence of the digital subscription platform capabilities, including payments, product catalog, customer accounts, financial accounting.
- Easing access to business data.
- Increasing the use of cloud-native technologies and AWS managed services.
- AWS Backup for Amazon EFS volumes.

Lessons Learned

The most significant lesson learned in the project was around testing, which ended up being the most time-consuming and underestimated part of the project by far (70-80 percent of the time). Test cases need to be granular enough and automated.

With the mainframe in operation for more than 35 years, the COBOL application had accumulated a fair amount of obsolete code due to a lack of adequate maintenance. It's a best practice to identify this code and remove it, which reduces the amount of refactoring and testing work to do.

Mainframes are typically backend processing systems for other servers. Aristo generated more than 3,500 data file feeds and reports for downstream consumers daily. Having a good inventory of all the consumers and interfaces facilitates the modernization and harmonization of the communications.

For Java code maintenance or new feature developments, it's a good practice to cross-train existing mainframe COBOL developers with Java skills. This allows them to provide both functional insight and knowledge about specific coding standards for the application, such as naming conventions or overall code structure.

Gaining a deep application understanding during the analysis and planning phase is important in order to define work packets, which are about the same size and complexity and allow reusing learnings for later packets. For example, it's good to start with migrating the batch jobs that are often of high complexity and demanding.

If The New York Times had its Cloud strategy already in place before starting the mainframe migration, the company would have chosen to migrate the mainframe directly to AWS, avoiding the extra work for designing and implementing the on-premises Aristo deployment.

Automated Refactoring of a New York Times Mainframe to AWS with Advanced

Project Benefits for The New York Times

While the modernization project began as a cost-cutting exercise, ultimately The New York Times took methodical and incremental steps toward more cutting edge technology adoption. All of this was done in an effort to improve customer service and gain competitive advantage in a unique industry that has seen significant market dynamic shifts since the project first began in 2015.

This project allowed convergence on a common technology stack (Java and Oracle on AWS) joining the Digital Subscription Platform that is now run, built, and maintained by the same Subscription Platforms group within The New York Times Technology organization.

The team is accelerating how it builds software on the new platform by adopting an agile methodology and CI/CD pipeline. In addition, there is now easier access to data gaining business and technology insights, and more rapid use of cloud-native technologies.

Aristo went live on August 28, 2017. During the first year, it has billed over half a billion dollars in subscription revenue, processed nearly 6.5 million transactions, and continued to route the daily paper to The New York Times' home delivery subscribers across the United States.

Remarkably, Aristo today costs 70% less to operate per year than it did to run on the mainframe in 2015, giving The New York Times a significant cost savings.

More information

w modernsystems.oneadvanced.com **e** hello@oneadvanced.com

UK +44 0333 230 1884

Ditton Park, Riding Court Road Datchet, Slough, Berkshire, SL3 9LL

US +1 855-905-4040

3200 Windy Hill Road, Suite 230 West, Atlanta, GA 30339